Introducción al diseño de procesadores.

¿Qué necesito para comenzar el diseño de un procesador?
Primero vamos a establecer las instrucciones básicas que va a manejar. Estas instrucciones se muestran en la tabla 1.

Instrucciones de carga y almacenamiento			
Instr.	Sintaxis	Significado	Tipo
LD	LD ACCx, #n	ACCx = n	I
	LD ACCx, dir	ACCx = [dir]	D
ST	ST ACCx, dir	[dir] = ACCx	D
Instrucciones aritméticas			
ADD	ADD ACCx	ACCx = ACCA+ACCB	R
	ADD ACCx, #n	ACCx = ACCx+n	I
	ADD ACCx, dir	ACCx = ACCx+[dir]	D
SUB	SUB ACCx	ACCx = ACCA - ACCB	R
	SUB ACCx, #n	ACCx = ACCA - n	I
	SUB ACCx, dir	ACCx = ACCA - [dir]	D
Instrucciones de salto incondicional			
B	B #n	PC = #n	I

Tabla 1: Instrucciones del ESCOMICRO1

A estas instrucciones se le conoce como **“Conjunto de instrucciones del lenguaje ensamblador”**. Los lenguajes ensambladores son diferentes de un microprocesador a otro, sin embargo, las funciones que realizan cada uno de ellos son las mismas, es decir, todos los lenguajes ensambladores tienen instrucciones de carga y almacenamiento, aritméticas y lógicas, de salto incondicional, etc.

Con estas instrucciones podemos realizar diversos programas. Vamos a realizar dos ejercicios.

Ejercicio 1: Hacer un programa que sume dos números (7, 12) y el resultado se coloque en la dirección de memoria A0H.

Versión 1	Significado	Versión 2	Significado

LD ACCA, #7
LD ACCB, #12
ADD ACCA
ST ACCA, A0H
	ACCA = 7
	ACCB = 12
	ACCB = ACCA + ACCB
	[A0H] = ACCB
		LD ACCA, #7
		ADD ACCA, #12
		ST ACCA, A0H
			ACCA = 7
			ACCA = ACCA + 12
			[A0H] = ACCA

Ejercicio 2: Hacer un programa que inicie el acumulador A en 7 (A = 7) y lo incremente de uno en uno (A + 1), el resultado se debe colocar en la dirección de memoria 90H. Hacer este código dentro de un ciclo infinito.

Versión 1	Significado	Versión 2	Significado

LD ACCA, #7
LD ACCB, #1
CICLO:
ADD ACCA
ST ACCA, 90H
B CICLO
	ACCA = 7
	ACCB = 1
	
	ACCA = ACCA + ACCB
	[90H] = ACCA
		LD ACCA, #7
		CICLO:
		ADD ACCA, #1
		ST ACCA, 90H
		B CICLO
			ACCA = 7
			
			ACCA = ACCA + 1
			[90H] = ACCA

¿Cómo vamos a representar esas instrucciones para que puedan ser ejecutadas?

Para poder representarlas necesitamos definir un **formato para las instrucciones**. Este formato debe representar el tipo de instrucción (carga y almacenamiento, aritméticas, salto incondicional, etc.), sus operandos (Acumulador A ó B) y el modo de direccionamiento (tipo R, I, D). El **modo de direccionamiento** nos va a decir cuales son los operandos que puedo manejar con una instrucción del ensamblador, por ejemplo, una instrucción de suma (ADD), puede sumar dos registros, un registro y un número inmediato, un registro y una dirección de memoria (variable).

Un formato que permite representar todo esto se muestra en la figura 1:

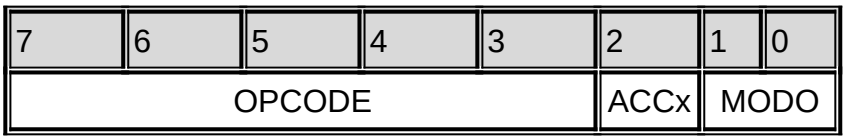

Figura 1: Formato de instrucción.

OPCODE. Cada formato de instrucción contiene un código de operación (OPCODE) que permite identificar una instrucción de otra. Este código esta formado por 5 bits que se encuentran de los bits 7 al 3. Con estos 5 bits podemos tener hasta 32 instrucciones diferentes en nuestro procesador.

ACCx. El bit 2 indica el acumulador que va a utilizar la instrucción para colocar el resultado. Bit2 con 1 significa acumulador B, con 0 significa acumulador A.

MODO. Los bits 1 y 0 permiten especificar el modo de direccionamiento usado en las instrucciones.

Modo	Bit 1	Bit 0
Tipo R	0	0
Tipo I	0	1
Tipo D	1	0
Tipo Ind	1	1

Tabla 2: Modos de direccionamiento

Con este formato de 8 bits podemos obtener los formatos específicos para cada modo de direccionamiento.

Formato Tipo R (Registro). No existe operando. Los operandos a usar en la instrucción se encuentran en los registros acumuladores (ACCA, ACCB). Esto quiere decir que previamente fueron almacenados los operandos en los registros acumuladores. **Solo se usa un byte**. El formato se muestra en la figura 2.

7	6	5	4	3	2	1	0
OPCODE					ACCx	0	0

Figura 2: Formato de instrucción Tipo R.

Tipo I (Inmediato). Se usa una byte para el formato de la instrucción y otro para el dato inmediato. Este formato se muestra en al figura 3.

7	6	5	4	3	2	1	0
OPCODE					ACCx	0	1

7	6	5	4	3	2	1	0
DATO INMEDIATO							

Figura 3: Formato de instrucción Tipo I.

Tipo D (Directo). Se usa un byte para el formato de la instrucción y otro para la dirección de memoria donde se encuentra el dato a usar. Este formato se muestra en al figura 4.

7	6	5	4	3	2	1	0
OPCODE					ACCx	1	0

7	6	5	4	3	2	1	0
DIRECCIÓN							

Figura 4: Formato de instrucción Tipo D.

Con estos formatos de instrucción podemos representar las instrucciones de la tabla 1 con un código binario, tal como se muestra en la tabla 3.

Instrucciones de carga y almacenamiento				
Instrucción	Modos de direccionamiento	Significado	Formato de la instrucción	Tipo
LD	LD ACCx, #n	ACCx = n	00000 A 01 nnnn nnnn	I
	LD ACCx, dir	ACCx = [dir]	00000 A 10 dddd dddd	D
ST	ST ACCx, dir	[dir] = ACCx	00001 A 10 dddd dddd	D
Instrucciones aritméticas				
ADD	ADD ACCx	ACCx = ACCA+ACCB	00010 A 00	R
	ADD ACCx, #n	ACCx = ACCx+n	00010 A 01 nnnn nnnn	I
	ADD ACCx, dir	ACCx = ACCx+[dir]	00010 A 10 dddd dddd	D
SUB	SUB ACCx	ACCx = ACCA - ACCB	00011 A 00	R
	SUB ACCx, #n	ACCx = ACCA - n	00011 A 01 nnnn nnnn	I
	SUB ACCx, dir	ACCx = ACCA - [dir]	00011 A 10 dddd dddd	D
Instrucciones de saltos				
B	B #n	PC = #n	00100 x 01 nnnn nnnn	I

Tabla 3: Formato de las instrucciones del ESCOMICRO1.

Con estos formatos de instrucción podemos representar las instrucciones de los programas del ejemplo1 en la memoria ROM, mejor conocida como “memoria de programa”, tal como se muestra en las tablas 4 y 5.

Memoria ROM (Memoria de programa)

	OPCODE					ACCx	MODO		
Dir	D7	D6	D5	D4	D3	D2	D1	D0	Instr
0	0	0	0	0	0	0	0	1	LD ACCA, #7
1	0	0	0	0	0	1	1	1	
2	0	0	0	0	0	1	0	1	LD ACCB, #12
3	0	0	0	0	1	1	0	0	
4	0	0	0	1	0	0	0	0	ADD ACCA
5	0	0	0	0	1	0	1	0	ST ACCA, A0H
6	0	0	0	0	0	1	1	1	

Tabla 4: Ejercicio 1 en su versión 1.

Memoria ROM (Memoria de programa)

	OPCODE					ACCx	MODO		
Dir	D7	D6	D5	D4	D3	D2	D1	D0	Instr
0	0	0	0	0	0	0	0	1	LD ACCA, #7
1	0	0	0	0	0	1	1	1	
2	0	0	0	1	0	1	0	1	ADD ACCA, #12
3	0	0	0	0	1	1	0	0	
5	0	0	0	0	1	0	1	0	ST ACCA, A0H
6	0	0	0	0	0	1	1	1	

Tabla 5: Ejercicio 1 en su versión 2.

Las instrucciones de los programas del ejemplo 2 se representan con sus formatos en la memoria de programa tal como se muestra en las tablas 6 y 7.

Memoria ROM (Memoria de programa)

	OPCODE					ACCx	MODO		
Dir	D7	D6	D5	D4	D3	D2	D1	D0	Instr
0	0	0	0	0	0	0	0	1	LD ACCA, #7
1	0	0	0	0	0	1	1	1	
2	0	0	0	0	0	1	0	1	LD ACCB, #1
3	0	0	0	0	0	0	0	1	
4	0	0	0	1	0	0	0	0	CICLO: ADD ACCA
5	0	0	0	0	1	0	1	0	ST ACCA, 90H
6	1	0	0	1	0	0	0	0	
7	0	0	1	0	0	X	0	1	B CICLO
8	0	0	0	0	0	1	0	0	

Tabla 6: Ejercicio 2 en su versión 1.

Memoria ROM (Memoria de programa)

	OPCODE					ACCx	MODO		
Dir	D7	D6	D5	D4	D3	D2	D1	D0	Instr
0	0	0	0	0	0	0	0	1	LD ACCA, #7
1	0	0	0	0	0	1	1	1	
2	0	0	0	1	0	0	0	1	CICLO: ADD ACCA, #1
3	0	0	0	0	0	0	0	1	
4	0	0	0	0	1	0	1	0	ST ACCA, 90H
5	1	0	0	1	0	0	0	0	
6	0	0	1	0	0	X	0	1	B CICLO
7	0	0	0	0	0	0	1	0	

Tabla 7: Ejercicio 2 en su versión 2.

El conjunto de instrucciones y su formato de instrucción, mostrados en la tabla 3, dan como resultado la arquitectura mostrada en la figura 5.

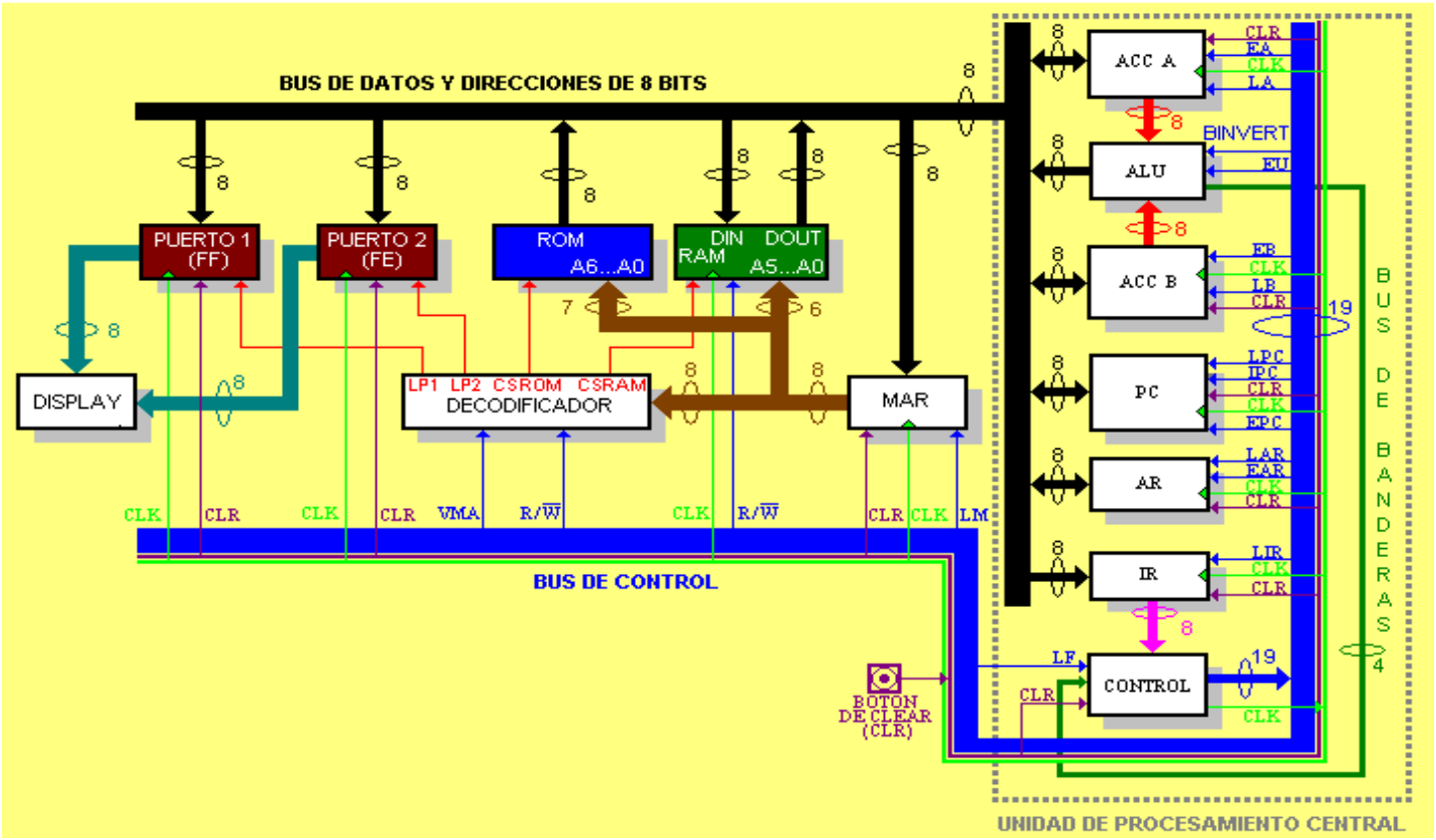

Figura 5: Arquitectura del ESCOMICRO1

Autor: Victor Hugo García Ortega

Este microprocesador lo llamamos el ESCOMICRO1 y esta formado de los bloques funcionales siguientes:

Acc A, Acc B (Acumuladores A y B). Los registros acumuladores tiene los datos con los que opera directamente la ALU. Manejan las señales LA, LB que permiten controlar la carga de datos. También las señales EA, EB que coloca el bus de salida de los acumuladores en alta impedancia o permiten habilitarlos para colocar su dato almacenado en el bus de datos y direcciones.

LA, LB	Operación
0	Retención
1	Carga

EA, EB	Operación
0	Alta impedancia
1	Habilitación

ALU (Unidad Aritmética y Lógica). Se encarga de las operaciones básicas de suma y resta. Maneja la señal Binvert que permite seleccionar la operación de suma o resta de los valores que contengan los acumuladores. También la señal EU que coloca el bus de salida de la ALU en alta impedancia o permite colocar su dato en el bus de datos y direcciones.

BINVERT	Operación
0	Suma (ACCA + ACCB)
1	Resta (ACCA - ACCB)

EU	Operación
0	Alta impedancia
1	Habilitación

PC (Contador de Programa). El PC tiene la dirección de la siguiente instrucción a ejecutar por el microprocesador. Su función es la de emitir direcciones de memoria. Maneja la señal IPC que permite habilitar el conteo ascendente desde 0 hasta 255 (8 bits). También la señal EPC que coloca el bus de salida del PC en alta impedancia o permite colocar su dirección en el bus de datos y direcciones.

IPC	Operación
0	Retención
1	Conteo Ascendente

EPC	Operación
0	Alta impedancia
1	Habilitación

IR (Registro de Instrucciones). Este registro es donde se tiene que almacenar la instrucción tomada de memoria durante la fase de búsqueda de la instrucción denominada FETCH. Manejan la señal LIR que permite controlar la carga de la instrucción a ejecutar.

LIR	Operación
0	Retención
1	Carga

AR (Registro Auxiliar). Los registros auxiliares, siempre numerosos en cualquier microprocesador. Este registro sirve como depósito momentáneo de los datos usados en las microinstrucciones usadas en las instrucciones tipo D. Manejan la señal LAR que permite controlar la carga de datos. También la señales EAR que coloca el bus de salida del registro en alta impedancia o permite habilitarlo para colocar su dato almacenado en el bus de datos y direcciones.

LAR	Operación
0	Retención
1	Carga

EAR	Operación
0	Alta impedancia
1	Habilitación

MAR. (Registro de Direcciones de Memoria). Este registro es necesario sólo cuando nos encontremos en presencia de un microprocesador con un solo BUS para datos y direcciones (compartido). Con este tipo de bus, resulta evidente que no es posible usar el mismo BUS al mismo tiempo para los datos y para las direcciones. El registro MAR resuelve exactamente este problema ya que almacena la dirección que se va acceder por las memorias. Manejan la señal LM que permite controlar la carga de la dirección a decodificar.

LM	Operación
0	Retención
1	Carga

Decodificador

Decodifica las direcciones de memoria para activar las señales de habilitación (CS) de cada memoria dependiendo de los rangos de direcciones asignadas a la memoria ROM, RAM o periféricos. También habilita la carga en los registros que funcionan como puertos de salida. Maneja la señal WMA (Valid Memory Address) que permite establecer cuando una dirección dentro del registro MAR es válida para su decodificación. También maneja la señal RW que permite seleccionar una operación de lectura o escritura en las memorias ROM o RAM y en los puertos.

VMA	Operación
0	Dirección no válida
1	Dirección válida

RW	Operación
0	Escritura
1	Lectura

Memoria ROM

Esta memoria se le denomina memoria de programa y es la encargada de contener las instrucciones a ejecutar por el microprocesador, esta mapeada en la parte baja del mapa de memoria en el rango de direcciones de 00H-7FH. Maneja un bus de direcciones de 7 bits (A6, …, A0) y un bus de datos de 8 bits (D7, …, D0), por lo que su organización es de 128 x 8.

Memoria RAM

Esta memoria se le denomina memoria de datos y es la encargada de contener las variables usadas por las instrucciones del ensamblador en el microprocesador, esta mapeada en la parte media baja del mapa de memoria en el rango de direcciones de 80H-BFH. Maneja un bus de direcciones de 6 bits (A5, …, A0) y un bus de datos de 8 bits (D7, …, D0), por lo que su organización es de 64 x 8.

Puertos

Son dos registros mapeados en la parte alta del mapa de memoria, en las direcciones FFh y FEh, y están diseñados para manejar dos puertos de 8 bits de salida para el manejo de periféricos. Manejan las señales LP1, LP2 que permiten controlar la carga de datos.

LP1, LP2	Operación
0	Retención
1	Carga

Unidad de Control.

Es la encargada de realizar la decodificación de la instrucción, es decir, analizar el código de operación para determinar la instrucción, el acumulador usado como operando y el tipo de modo de direccionamiento. Una vez decodificada la instrucción genera las microinstrucciones en las fases de ejecución adecuadas para ejecutar cada instrucción del ensamblador. Este bloque activa TODAS las señales de control de cada unidad funcional del procesador (Acumuladores, ALU, PC, IR, AR, MAR, Decodificador, Puertos y Memorias).

Las señales de control de cada bloque funcional puede tener una acción síncrona o asíncrona.

Las señales tienen una acción síncrona cuando se lleva a cabo en el momento que llega el flanco activo de la señal de reloj. Todas las señales de control etiquetadas como LA, LB, LPC, IPC, LAR, LM, LIR, RW (en la memoria RAM) son síncronas.

Las señales tienen una acción asíncronas cuando se lleva a cabo sin la necesidad de una señal de reloj. Las señales EA, EB, EU, EPC, EAR, VMA, RW (en el decodificador) son señales asíncronas.